Universally Composable DKG with Linear Number of Exponentiations

Douglas Wikström

Royal Institute of Technology (KTH)
KTH, Nada, S-100 44 Stockholm, Sweden

26th May 2004

Abstract. Many problems have been solved by protocols using discrete-logarithm based threshold cryptosystems. Such protocols require a random joint public key for which the secret key is shared among the parties. A multiparty protocol that generates such a key is called a DKG protocol. Until now no DKG protocol is known to be universally composable.

We extend Feldman's original verifiable secret sharing scheme to construct a DKG protocol, and prove that it is universally composable. Our result holds in a common random string model under the Decision Diffie-Hellman assumption. We stress that we do not need any trapdoor for the common random string.

Our protocol is optimistic. If all parties behave honestly, each party computes only $O(3k)$ exponentiations, where k is the number of parties. In the worst case each party computes $O(k^2)$ exponentiations. This should be contrasted with previous constructions in which each party computes $\Omega(k^2)$ exponentiations regardless of if they behave honestly or not. In the optimistic case the number of bits sent in our protocol is essentially equal to the number of bits sent in k independent copies of Feldman's original protocol.

1 Introduction

The ability of a group of parties to jointly generate a public key for which the secret key is shared is a cornerstone of threshold cryptography. Without a method to do this securely the parties must resort to a preliminary phase in which a trusted key generating party is present. In some applications no natural trusted party exists, e.g. electronic voting.

When a discrete-logarithm based cryptosystem is used, distributed key generation amounts to generating a public key $y = g^x$ for which the corresponding secret key x is secretly and verifiably shared among the parties. Following Gennaro et al. [13] we call a protocol that securely realizes such a functionality a DKG protocol.

1.1 Previous Work

The problem of constructing a DKG protocol was first investigated by Pedersen [25]. His basic building block was a new non-interactive verifiable secret sharing scheme [24] based on ideas of Feldman [7]. Pedersen DKG has been used as a subprotocol in numerous constructions in the literature, but it has never been verified that Pedersen DKG composes correctly in general. Indeed, Gennaro et al. [13] pointed out that the Pedersen DKG may generate a public key which is biased by the adversary. They also gave a new modified protocol and gave a more careful analysis. Adaptively secure protocols for DKG where given by Canetti et al. [4] and Jarecki and Lysyanskaya [20]. Independently, Frankel, MacKenzie and Yung gave key generation protocols secure against adaptive adversaries in several papers [8,9,10,11]. They also considered threshold variants of RSA. Recently, Gennaro et al. [12] investigated the security of the original Pedersen DKG, i.e. how the adversary can benefit from biasing the public key. They show that under certain circumstances the adversary gains very little from this additional power.

Canetti [3] and independently Pfitzmann and Waidner [26], proposed security frameworks for reactive processes. We use the former framework. Both frameworks have composition theorems, and are based on older definitional work. The initial ideal-model based definitional approach for secure function evaluation is informally proposed by Goldreich, Micali, and Wigderson in [16]. The first formalizations appear in Goldwasser and Levin [17], Micali and Rogaway [23], and Beaver [1]. Canetti [2] presents the first definition of security that is preserved under composition. See [2,3] for an excellent background on these definitions.

1.2 Contribution

No existing DKG protocol is known to be universally composable. In this work we rectify this and give a protocol that securely realizes the ideal DKG functionality in a universally composable way under the Decision Diffie-Hellman assumption. This means that our protocol can be plugged as a subprotocol in any scenario where a DKG protocol is needed. Our result holds in a very weak common random string model. We do not need any trapdoor for our analysis.

Let k be the number of parties. If all parties behave honestly, each party computes only $O(3k)$ exponentiations and the number of bits sent in the protocol is essentially equal to the number of bits sent in k independent copies of Feldman's original protocol. In the worst case each party computes $O(k^2)$ exponentiations. Previous constructions require $\Omega(k^2)$ exponentiations regardless if all parties behave honestly or not.

Interestingly, we do not use Pedersen commitments [25] at any point in our protocol. Instead we use Feldman's original protocol in a novel way.

1.3 Notation

Throughout, $M_1, \ldots, M_k$ denote the participating parties, which are modeled as interactive Turing machines. We abuse notation and use M_j to denote both the machines themselves and their identity. We write $\mathcal{M}_{k/2}$ to denote the set of *static* adversaries that can corrupt a minority of the parties.

We use the term "randomly" instead of "uniformly and independently at random". We assume that G_q is a group of prime order q with generator g for which the Decision Diffie-Hellman Assumption holds, e.g. a subgroup G_q of prime order q of $\mathbb{Z}_p^*$ for some $p = \kappa q + 1$. Informally the assumption says that it is hard to distinguish the distributions $(g^\alpha, g^\beta, g^{\alpha\beta})$ and $(g^\alpha, g^\beta, g^\gamma)$ when $\alpha, \beta, \gamma \in \mathbb{Z}_q$ are randomly chosen.

We review the El Gamal [6] cryptosystem employed in G_q. The private key x is generated by choosing $x \in \mathbb{Z}_q$ randomly. The corresponding public key is $y = g^x$. Encryption of a message $m \in G_q$ using the public key y is given by $E_y(m, r) = (g^r, y^r m)$, where r is chosen randomly from $\mathbb{Z}_q$, and decryption of a cryptotext on the form $(u, v) = (g^r, y^r m)$ using the private key x is given by $D_x(u, v) = u^{-x} v = m$. Tsionis and Yung [28] show that the El Gamal cryptosystem is semantically secure [19,22] under the DDH-assumption.

Throughout this paper we employ the universally composable security framework of Canetti [3]. Today this framework is well known and several flavors of it have evolved. Our result does not depend on technicalities of any particular flavor, but to avoid any ambiguity we review in Appendix B the definitions of the universally composable security we use in this paper.

The notion of a bulletin board is intuitively clear. Below follows a formal definition taken from [29].

Functionality 1 (Bulletin Board). The ideal *bulletin board* functionality, $\mathcal{F}_{\mathrm{BB}}$, running with parties $P_1, \ldots, P_k$ and ideal adversary $\mathcal{S}$.

1. $\mathcal{F}_{\mathrm{BB}}$ holds a database indexed on integers. Initialize a counter $c = 0$.
2. Upon receiving $(P_i, \mathtt{Write}, m_i)$, $m_i \in \{0,1\}^*$, from $\mathcal{C}_{\mathcal{I}}$, store (P_i, m_i) under the index c in the database, hand $(\mathcal{S}, \mathtt{Write}, c, P_i, m_i)$ to $\mathcal{C}_{\mathcal{I}}$, and set $c \leftarrow c+1$.
3. Upon receiving $(P_j, \mathtt{Read}, c)$ from $\mathcal{C}_{\mathcal{I}}$ check if a tuple (P_i, m_i) is stored in the database under c. If so hand $((\mathcal{S}, P_j, \mathtt{Read}, c, P_i, m), (P_j, \mathtt{Read}, c, P_i, m_i))$ to $\mathcal{C}_{\mathcal{I}}$. If not, hand $((\mathcal{S}, P_j, \mathtt{NoRead}, c), (P_j, \mathtt{NoRead}, c))$ to $\mathcal{C}_{\mathcal{I}}$.

Goldwasser and Lindell [18] show that an authenticated broadcast can be securely realized with respect to *blocking* $\mathcal{M}_{k/2}$-adversaries. On the other hand Lindell, Lysyanskaya and Rabin [21] show that composable authenticated broadcast can not be realized for *non-blocking* $\mathcal{M}_B$-adversaries if $B > k/3$. The following lemma follows straightforwardly from [18].

Lemma 1. *There exists a protocol π_{BB} that securely realizes $\mathcal{F}_{\mathrm{BB}}$ with respect to* blocking *$\mathcal{M}_{k/2}$-adversaries.*

In many constructions the parties are assumed to be able to communicate secretly with each other. In the UC-framework this was modeled by Canetti [3]. Below we give a slightly modified variant, better suited for our setting.

Functionality 2 (Multiple Message Transmission). The ideal multiple message transmission, $\mathcal{F}_{\mathrm{MMT}}$, running with parties $P_1, \ldots, P_k$ and ideal adversary $\mathcal{S}$.

1. In the first activation expect to receive a value (Receiver) from some party P_j. Then hand $((\mathcal{S}, \texttt{Receiver}, P_j), \{(P_i, \texttt{Receiver}, P_j)\}_{i=1}^k)$ to $\mathcal{C}_\mathcal{I}$.
2. Upon receiving $(P_j, \texttt{Send}, P_i, m_j)$ from $\mathcal{C}_\mathcal{I}$, hand $((\mathcal{S}, P_j, \texttt{Send}, P_i, |m_j|), (P_i, P_j, m_j))$ to $\mathcal{C}_\mathcal{I}$.

From Claim 16 in Canetti [3] and the fact that the Cramer-Shoup cryptosystem [5] is chosen ciphertext secure in the sense of Rackoff and Simon [27] under the Decision Diffie-Hellman assumption in G_q, the following lemma follows straightforwardly.

Lemma 2. *There exists a protocol π_{MMT} that securely realizes $\mathcal{F}_{\mathrm{MMT}}$ under the Decision Diffie-Hellman assumption in G_q.*

A common assumption used in the construction of protocols is the existence of a common random string.

Functionality 3 (Common Random String (CRS)). The ideal common random string, $\mathcal{F}_{\mathrm{CRS}}$, running with parties $P_1, \ldots, P_k$ and ideal adversary $\mathcal{S}$.

1. Choose $h_1, h_2, h_3 \in G_q$ randomly. Then hand $((\mathcal{S}, \texttt{CRS}, h_1, h_2, h_3), \{(P_j, \texttt{CRS}, h_1, h_2, h_3)\}_{j=1}^k)$ to $\mathcal{C}_\mathcal{I}$.

The notion of a common reference string is different from the above in that the common string may have some additional structure, i.e. it may be an RSA modulus. Sometimes a hidden trapdoor is also used in simulations, but we need neither structure nor trapdoors.

2 Distributed Key Generation

The functionality below captures the notion defined by Gennaro et al. [13], but in the language of the UC-framework. A public key $y = g^x$ is generated and given to all parties. Each party also receives a share of the secret key x.

Functionality 4 (Distributed Key Generation (DKG)). The ideal *Distributed Key Generation over G_q*, $\mathcal{F}_{\mathrm{KG}}$, running with generators $M_1, \ldots, M_k$, and ideal adversary $\mathcal{S}$ proceeds as follows. Let $t = \lceil k/2 - 1 \rceil$.

1. Wait for $(\texttt{CorruptShares}, \{j, s_j\}_{j \in I_M})$ from $\mathcal{S}$, where I_M is the set of indices of corrupted parties.
2. Choose $a_\iota \in \mathbb{Z}_q$ randomly under the restriction $a(j) = s_j$ for $j \in I_M$, where $a(z) = \sum_{\iota=0}^t a_\iota z^\iota$. Then define $s_j = a(j)$ for $j \notin I_M$, and set $y = g^{a_0}$.
3. Hand $((\mathcal{S}, \texttt{PublicKey}, y), \{(\texttt{PublicKey}, y, s_j)\}_{j=1}^k)$ to $\mathcal{C}_\mathcal{I}$.

Note that the adversary may choose the shares handed to corrupted parties. This obviously gives no additional information on the shared secret, but it is convenient for technical reasons.

Before we give our protocol, recall the verifiable secret sharing scheme of Feldman [7]. A dealer shares a secret $a_0 \in \mathbb{Z}_q$ by choosing $a_\iota \in \mathbb{Z}_q$ randomly and forming a polynomial $a(z) = \sum_{\iota=0}^{t} a_\iota z^\iota$ of degree t. Then it publishes $\alpha_\iota = g^{a_\iota}$ and hands a share $s_j = a(j)$ to the j:th party. This allows the receiver of a share s_j to verify its correctness by checking that $g^{s_j} = \prod_{\iota=0}^{t} \alpha_\iota^{j^\iota}$. If a share is not correct the receiver complains and forces the dealer to publish a correct share. To recover the secret the receivers simply publish their shares. This allows anybody to find a set of correct shares and Lagrange interpolate the secret. The distribution step of Feldman's protocol gives a way to share the secret key x corresponding to a public key $\alpha_0 = g^x$.

It may appear that one could let each party run a copy of Feldman's protocol and then multiply all public keys to construct a joint public key, but Gennaro et al. [13] show that if this is done the adversary may bias the distribution of the joint public key.

Next we give an informal description of our protocol. To start with the parties are partitioned into three groups and each group is assigned a random generator $h_f \in G_q$. Each party runs a copy of Feldman's protocol, but using its group's generator. Instead of verifying each individual dealer's shares and public information, they are combined within each group of parties. This allows efficient verification. The basic idea behind this trick was taken from Gennaro et al. [14]. If some party is malicious, the efficient way of verifying shares is abandoned and individual verifications are performed. From this each party M_j computes a combined share s_j, the sum of all correct shares it received. Then each party publish $\beta_j = g^{s_j}$. Note that this time all parties use the generator g. Then the parties verify the correctness of the β_j's and construct a joint key $y = g^x$, for which x is the secret to which the combined shares s_j correspond. If the verification fails each party is essentially required to prove that its β_j is correct. Because of the structure of the protocol this can be done by simply publishing some elements from G_q, i.e. no secret communication takes place. This allows all parties to agree on a set of correct β_j from which the joint key can be constructed.

Let $\{\Omega_1, \Omega_2, \Omega_3\}$ be a partition of $\{1, \ldots, k\}$ such that $||\Omega_f| - |\Omega_{f'}|| \leq 1$ for $f \neq f'$. Define $f(j)$ to be the value of f such that $j \in \Omega_f$.

Protocol 1 (Distributed Key Generation (DKG)). Let $t = \lceil k/2 - 1 \rceil$. The Distributed Key Generation protocol $\pi = (M_1, \ldots, M_k)$ consists of generators M_j. Each generator M_j proceeds as follows.

Preliminary Phase

1. Hand $(\texttt{Receiver})$ to $\mathcal{F}_{\mathrm{MMT}}$.
2. Wait for $(\texttt{Receiver}, M_l)$ for $l \neq j$ from $\mathcal{F}_{\mathrm{MMT}}$.
3. Wait for $(\texttt{CRS}, g, h_1, h_2, h_3)$ from $\mathcal{F}_{\mathrm{CRS}}$.

Key Generation Phase

4. Define f by $j \in \Omega_f$. Choose $a_{j,\iota} \in \mathbb{Z}_q$ randomly, and define

$$a_j(z) = \sum_{\iota=0}^{t} a_{j,\iota} z^\iota, \quad \alpha_{j,\iota} = h_f^{a_{j,\iota}}, \quad \text{and} \quad s_{j,l} = a_j(l) \ .$$

Hand $(\texttt{Send}, M_l, \texttt{Share}, s_{j,l})$ to $\mathcal{F}_{\mathrm{MMT}}$ for $l \neq j$ and $(\texttt{Write}, \texttt{PublicElements}, \{\alpha_{j,\iota}\}_{\iota=0}^{t})$ to $\mathcal{F}_{\mathrm{BB}}$.

5. Wait until $(M_l, \texttt{PublicElements}, \{\alpha_{l,\iota}\}_{\iota=1}^{t})$ appears on $\mathcal{F}_{\mathrm{BB}}$ and $(P_l, \texttt{Share}, s_{l,j})$ is received from $\mathcal{F}_{\mathrm{MMT}}$ for $l = 1, \ldots, k$. Verify that

$$h_f^{\sum_{l \in \Omega_f} s_{l,j}} = \prod_{\iota=0}^{t} \Big(\prod_{l \in \Omega_f} \alpha_{l,\iota} \Big)^{j^\iota} .$$

for $f = 1, 2, 3$. If so, hand $(\texttt{Write}, \texttt{Complaints}, \emptyset)$ to $\mathcal{F}_{\mathrm{BB}}$. Otherwise, go to Step 10.

6. Wait until $(M_l, \texttt{Complaints}, \Delta_l)$ appears on $\mathcal{F}_{\mathrm{BB}}$ for $l \neq j$. If all $\Delta_l = \emptyset$ set $I_1 = \{1, \ldots, k\}$. Otherwise, go to Step 11.
7. Define $s_j = \sum_{l \in I_1} s_{l,j}$ and set $\beta_j = g^{s_j}$. Then hand $(\texttt{Write}, \texttt{ConstructPublicKey}, \beta_j)$ to $\mathcal{F}_{\mathrm{BB}}$.
8. Wait until $(M_l, \texttt{ConstructPublicKey}, \beta_l)$ appears on $\mathcal{F}_{\mathrm{BB}}$ for $l \in I_1$. Verify that:

$$\beta_1 = \prod_{i=1}^{t+1} \beta_i^{\prod_{l \neq i} \frac{l-1}{l-i}} .$$

If so set $I_2 = \{1, \ldots, t+1\}$. If not, go to Step 13.

9. Define

$$y = \prod_{i \in I_2} \beta_i^{\prod_{l \neq i} \frac{l}{l-i}} .$$

and output $(\texttt{PublicKey}, y, s_j)$.

Handle Cheating with the $s_{l,i}$ and $\alpha_{l,\iota}$.

10. Verify for $l = 1, \ldots, k$ that

$$h_{f(l)}^{s_{l,j}} = \prod_{\iota=0}^{t} \alpha_{l,\iota}^{j^\iota} ,$$

Let Δ_j be the set of indices l for which equality does not hold. Then hand $(\texttt{Write}, \texttt{Complaints}, \Delta_j)$ to $\mathcal{F}_{\mathrm{BB}}$.

11. Wait until $(M_l, \texttt{Complaints}, \Delta_l)$ appears on $\mathcal{F}_{\mathrm{BB}}$ for $l \neq j$. Let $\Gamma_j = \{l \mid j \in \Delta_l\}$. Then hand $(\texttt{Write}, \texttt{Refutes}, \{s_{j,l}\}_{l \in \Gamma_j})$ to $\mathcal{F}_{\mathrm{BB}}$.
12. Wait until $(M_l, \texttt{Refutes}, \{s_{l,i}\}_{i \in \Gamma_l})$ appears on $\mathcal{F}_{\mathrm{BB}}$ for $l \neq j$ and replace old values of $s_{l,j}$ with the new. Let I_1 be the set of l such that

$$h_{f(l)}^{s_{l,i}} = \prod_{\iota=0}^{t} \alpha_{l,\iota}^{i^\iota} ,$$

for $l = 1, \ldots, k$ and $i \in \{j\} \cup \bigcup_{l=1}^{k} \Gamma_l$. Then go to Step 7.

Handle Cheating with the β_l.

13. Set $c_{j,0} = s_j$ and choose $c_{j,\iota} \in \mathbb{Z}_q$ for $\iota > 0$ randomly. Define $c_j(z) = \sum_{\iota=0}^{t} c_{j,\iota} z^\iota$, $\gamma_{j,\iota} = g^{c_{j,\iota}}$, $\delta_{j,\iota} = h_{f(j)}^{c_{j,\iota}}$, and $\zeta_{j,l} = c_j(l)$.
    Then hand $(\texttt{Send}, P_l, \texttt{Share2}, \zeta_{j,l})$ to $\mathcal{F}_{\mathrm{MMT}}$ and hand $(\texttt{Write}, \texttt{PublicElements2}, \{\gamma_{j,\iota}, \delta_{j,\iota}\}_{\iota=1}^{t})$ to $\mathcal{F}_{\mathrm{BB}}$.
14. Wait until $(M_l, \texttt{PublicElements2}, \{\gamma_{l,\iota}, \delta_{l,\iota}\}_{\iota=1}^{t})$ appears on $\mathcal{F}_{\mathrm{BB}}$ for $l \in I_1$. Set $\gamma_{l,0} = \beta_l$ and $\delta_{l,0} = \prod_{i \in I_1} \alpha_{i,\iota}$ Then verify that
    $$g^{\zeta_{l,j}} = \prod_{\iota=0}^{t} \gamma_{l,\iota}^{j^\iota}, \quad \text{and} \quad h_{f(l)}^{\zeta_{l,j}} = \prod_{\iota=0}^{t} \delta_{l,\iota}^{j^\iota} \ .$$
    Let Δ'_j be the set of indices for which equality does not hold. Then hand $(\texttt{Write}, \texttt{Complaints2}, \Delta'_j)$ to $\mathcal{F}_{\mathrm{BB}}$.
15. Wait until $(M_l, \texttt{Complaints2}, \Delta'_l)$ appears on $\mathcal{F}_{\mathrm{BB}}$ for $l \in I_1$. Let $\Gamma'_j = \{l \mid j \in \Delta'_l\}$. Then hand $(\texttt{Write}, \texttt{Refutes2}, \{\zeta_{j,l}\}_{l \in \Gamma'_j})$ to $\mathcal{F}_{\mathrm{BB}}$.
16. Wait until $(M_l, \texttt{Refutes2}, \{\zeta_{l,i}\}_{i \in \Gamma_l})$ appears on $\mathcal{F}_{\mathrm{BB}}$ for $l \in I_1$ and replace the old values of $\zeta_{l,j}$ with the new. Let I_2 be the lexicographically first set of l such that
    $$g^{\zeta_{l,i}} = \prod_{\iota=0}^{t} \gamma_{l,\iota}^{i^\iota}, \quad \text{and} \quad h^{\zeta_{l,i}} = \prod_{\iota=0}^{t} \delta_{l,\iota}^{i^\iota} \ ,$$
    for $l = 1, \ldots, k$ and $i \in \{j\} \cup \bigcup_{l=1}^{k} \Gamma_l$. Then go to Step 9.

In the protocol above, all shares from parties from a partition Ω_f are verified together. One can also consider verifying smaller sets together if some cheating is expected. Then if cheating is detected, only some of the shares have to be verified individually. This modification does not change the security analysis in any essential way. In Appendix A we prove the following theorem.

Theorem 1. *Protocol 1 securely realizes $\mathcal{F}_{\mathrm{KG}}$ with respect to $\mathcal{M}_{k/2}$-adversaries in the $(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{MMT}}, \mathcal{F}_{\mathrm{CRS}})$-hybrid model under the DDH-assumption.*

If all parties behave honestly, each party computes $O(3k)$ exponentiations in G_q. In the worst case each party computes $O(k^2)$ exponentiations.

The following result follows straightforwardly from Lemma 2 and Lemma 1 by use of the composition theorem of the UC-framework [3].

Corollary 1. *The composition of Protocol 1, and π_{MMT} securely realizes $\mathcal{F}_{\mathrm{KG}}$ with respect to $\mathcal{M}_{k/2}$-adversaries in the $(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{CRS}})$-hybrid model under the DDH-assumption. If composed with π_{BB} the result holds in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model for* blocking *adversaries.*

Some intuition behind the construction and for the proof of Theorem 1 follows. Since the adversary could potentially generate the shares it distributes *after* receiving shares from all honest parties, it can choose the shares it distributes such that the combined shares s_j of corrupted parties take on certain values. This is why we must allow the ideal adversary to do this as well in the DKG functionality. We stress that the adversary can not bias the resulting public key y in any way.

Already after the publishing of the public elements $\alpha_{j,\iota}$ the simulator can extract the secrets of corrupt parties, compute the resulting final combined shares, and feed them to the ideal functionality. To simulate honest dummy parties, the ideal adversary generates in the second step public elements $\beta_{j,\iota}$ that appear correct to the corrupt parties, but they are carefully chosen such that the public key y output by a protocol execution is identical to the public key output by the ideal functionality. This implies that the public key y does not correspond to the shares held by the simulated honest parties.

Intuitively, this is not a problem as long as the secret shares s_j are never revealed. One could imagine a slightly simpler protocol in which a single generator h, independent from g, was used in the first phase of the protocol. However, such a protocol would not be universally composable, since the environment has access to all shares s_j at the end of an execution and can verify these shares against the public elements $\alpha_{l,\iota}$. The ability to do this would allow the adversary to discover that it is executed as a blackbox.

The use of three independent generators h_1, h_2 and h_3 in the first phase ensures that no adversary or environment can check if the final shares s_j correspond to the public elements $\alpha_{l,\iota}$. From the Decision Diffie-Hellman assumption follows that it is hard to distinguish a triple $(h_1^a, h_2^b, a+b)$ from (h_1^a, h_2^b, c) for random $a, b, c \in \mathbb{Z}_q$. The protocol is arranged such that each combined share s_j essentially is a sum of the form $a + b$ for which only public elements $\alpha_{l,\iota}$ corresponding to h_1^a and h_2^b are known. Thus, to decide if it is run as a blackbox in a simulation the adversary must essentially decide if it is handed a DDH triple or not.

In the phase where cheating with the $\alpha_{l,\iota}$ is handled, the protocol essentially executes the verification step of k independent copies of Feldman's original protocol. In the phase where cheating with the β_l are handled, each party M_l essentially proves that its β_l is correct.

We stress that although the UC-framework allows it, we have not used any trapdoor for the common random string in any simulations. Thus, we feel that our use of the common random string is very mild.

3 Acknowledgments

I thank Johan Håstad for helpful discussions.

References

1. D. Beaver, *Foundations of secure interactive computation*, Crypto '91, LNCS 576, pp. 377-391, 1991.

2. R. Canetti, *Security and composition of multi-party cryptographic protocols*, Journal of Cryptology, Vol. 13, No. 1, winter 2000.
3. R. Canetti, *Universally Composable Security: A New Paradigm for Cryptographic Protocols*, http://eprint.iacr.org/2000/067 and ECCC TR 01-24. Extended abstract appears in 42nd FOCS, IEEE Computer Society, 2001.
4. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, *Adaptive security for threshold cryptosystems*, Crypto '99, LNCS 1666, pp. 98-115, 1999.
5. R. Cramer, V. Shoup, *A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack*, Crypto '98, pp. 13-25, LNCS 1462, 1998.
6. T. El Gamal, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, Vol. 31, No. 4, pp. 469-472, 1985.
7. P. Feldman, *A practical scheme for non-interactive verifiable secret sharing*, In Proceedings of the 28th FOCS, pages 427-438, 1987.
8. Y. Frankel, P. MacKenzie, M. Yung, *Adaptively-secure distributed threshold public key systems*, In Proceedings of 7th Annual European Symposium 1999, LNCS 1643, pp. 4-27, 1999.
9. Y. Frankel, P. MacKenzie, M. Yung, *Adaptively-secure optimal-resilience proactive RSA*, Asiacrypt '99, LNCS 1716, pp. 180-194, 1999.
10. Y. Frankel, P. MacKenzie, M. Yung, *Adaptive Security for the Additive-Sharing Based Proactive RSA*, Public Key Cryptography 2001, LNCS 1992, pp. 240-263, 2001.
11. Y. Frankel, P. MacKenzie, M. Yung, *Adaptively secure distributed public-key systems*, Theoretical Computer Science, Vol. 287, No. 2, September 2002.
12. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, *Secure Applications of Pedersen's Distributed Key Generation Protocol*, RSA Security '03, LNCS 2612, pp. 373-390, 2003.
13. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems*, Eurocrypt '99, LNCS 1592, pp. 295-310, 1999.
14. R. Gennaro, M. O. Rabin, T. Rabin, *Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography*, In Proceedings of the 1998 ACM Symposium on Principles of Distributed Computing, 1998.
15. O. Goldreich, *Foundations of Cryptography*, Cambridge University Press, 2001.
16. O. Goldreich, S. Micali, and A. Wigderson, *How to Play Any Mental Game*, 19th STOC, pp. 218-229, 1987.
17. S. Goldwasser, L. Levin, *Fair computation of general functions in presence of immoral majority*, Crypto '90, LNCS 537, pp. 77-93, 1990.
18. S. Goldwasser, Y. Lindell, *Secure Multi-Party Computation Without Agreement*, In Proceedings of the 16th DISC, LNCS 2508, pp. 17-32, 2002.
19. S. Goldwasser, S. Micali, *Probabilistic Encryption*, Journal of Computer and System Sciences (JCSS), Vol. 28, No. 2, pp. 270-299, 1984.
20. S. Jarecki, A. Lysyanskaya, *Adaptively Secure Threshold Cryptography without the Assumption of Erasure*, Eurocrypt 2000, LNCS 1807, 221-242, 2000.
21. Y. Lindell, A. Lysyanskaya, T. Rabin, *On the Composition of Authenticated Byzantine Agreement*, 34th STOC, pp. 514-523, 2002.
22. S. Micali, C. Rackoff, B. Sloan, *The notion of security for probabilistic cryptosystems*, SIAM Journal of Computing, Vol. 17, No. 2, pp. 412-426, 1988.
23. S. Micali, P. Rogaway, *Secure Computation*, Crypto '91, LNCS 576, pp. 392-404, 1991.

24. T. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, Crypto '91, pp. 129-140, 1991.
25. T. Pedersen, *A threshold cryptosystem without a trusted party*, Eurocrypt '91, pp. 129-140, 1991.
26. B. Pfitzmann, M. Waidner, *Composition and Integrity Preservation of Secure Reactive Systems*, 7th Conference on Computer and Communications Security of the ACM, pp. 245-254, 2000.
27. C. Rackoff, D. Simon, *Noninteractive zero-knowledge proofs of knowledge and chosen ciphertext attacks*,In Proceedings of the 22:nd ACM Symposium on Theory of Computing - STOC '89, pp. 433-444, 1991.
28. Y. Tsiounis, M. Yung, *On the Security of El Gamal based Encryption*, International workshop on Public Key Cryptography, LNCS 1431, pp. 117-134, 1998.
29. D. Wikström, *A Universally Composable Mix-Net*, Proceedings of First Theory of Cryptography Conference (TCC '04), LNCS 2951, pp. 315-335, 2004.

A Proof of Theorem 1

We construct an ideal adversary $\mathcal{S}$ that runs any hybrid adversary $\mathcal{A}$ as a black-box. Then we show that if $\mathcal{S}$ does not imply that the protocol is secure, the DDH-assumption is broken.

The Ideal Adversary $\mathcal{S}$. Let I_M be the set of indices of generators corrupted by $\mathcal{A}$. The ideal adversary $\mathcal{S}$ corrupts the dummy generators $\tilde{M}_j$ for $j \in I_M$. The ideal adversary is best described by starting with a copy of the original hybrid ITM-graph $(V, E) = \mathcal{Z}'(\mathcal{H}(\mathcal{A}, \pi^{\tilde{\pi}^{(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{MMT}}, \mathcal{F}_{\mathrm{CRS}})}}))$ where $\mathcal{Z}$ is replaced by a machine $\mathcal{Z}'$ that we define below. The adversary $\mathcal{S}$ simulates all machines in V except those in $\mathcal{A}'$, and the corrupted machines M_j for $j \in I_M$ under $\mathcal{A}'$:s control. We now describe how the machines M_j for $j \notin I_M$ are simulated.

Simulation of Links $(\mathcal{Z}, \mathcal{A})$, $(\mathcal{Z}, M_j)$ for $j \in I_M$. $\mathcal{S}$ simulates $\mathcal{Z}'$ and $\tilde{M}_j$ for $j \in I_M$, such that it appears as if $\mathcal{Z}$ and $\mathcal{A}$, and $\mathcal{Z}$ and M_j for $j \in I_M$ are linked directly.

1. When $\mathcal{Z}'$ receives m from $\mathcal{A}$, m is written to $\mathcal{Z}$, by $\mathcal{S}$. When $\mathcal{S}$ receives m from $\mathcal{Z}$, m is written to $\mathcal{A}$ by $\mathcal{Z}'$. This is equivalent to that $\mathcal{Z}$ and $\mathcal{A}$ are linked directly.
2. When $\mathcal{Z}'$ receives m from M_j for $j \in I_M$, m is written to $\mathcal{Z}$ by $\tilde{M}_j$. When $\tilde{M}_j$, $j \in I_M$, receives m from $\mathcal{Z}$, m is written to M_j by $\mathcal{Z}'$. This is equivalent to that $\mathcal{Z}$ and M_j are linked directly for $j \in I_M$.

Extraction from a Corrupt Generator. Note that since the last t generators M_j which distributes their shares $s_{j,l}$ may be corrupted, the adversary can choose $s_{j,l}$ for $j \in I_M$ such that $s_j = \sum_{l \in I_1} s_{l,j}$ takes on any values of its choice. This is why the ideal functionality $\mathcal{F}_{\mathrm{KG}}$ must allow the ideal adversary $\mathcal{S}$ to bias the shares given to the corrupted parties by it. Thus the ideal adversary $\mathcal{S}$ must somehow extract the s_j for $j \in I_M$ and then hand them to $\mathcal{F}_{\mathrm{KG}}$ to ensure that the shares s_j for $j = 1, \ldots, k$ that is eventually delivered to the environment are consistent with the public key.

When $(\texttt{Write}, \texttt{Complaints}, \Delta_j)$ has appeared on $\mathcal{F}_{\mathrm{BB}}$ for $j = 1, \ldots, k$, interrupt the simulation of $\mathcal{F}_{\mathrm{BB}}$. There are two cases. If there were no complaints,

$$h_f^{\sum_{l\in\Omega_f} s_{l,j}} = \prod_{\iota=0}^{t} \Big(\prod_{l\in\Omega_f} \alpha_{l,\iota} \Big)^{j^\iota} .$$

is satisfied for $j \notin I_M$ and $f = 1, 2, 3$. This implies that we can Lagrange interpolate

$$b_{f,i} = \sum_{l\in\Omega_f} s_{l,i} = \sum_{j=1}^{t+1} \Big(\sum_{l\in\Omega_f} s_{l,j} \Big) \prod_{l\neq j} \frac{i-l}{j-l} \tag{1}$$

for $f = 1, 2, 3$ and compute $s_i = b_{1,i} + b_{2,i} + b_{3,i}$ for $i \in I_M$. If there were complaints, we have for $l \in I_1$ and $j \notin I_M$

$$h_{f(l)}^{s_{l,j}} = \prod_{\iota=0}^{t} \alpha_{l,\iota}^{j^\iota} .$$

This implies that Equation (1) holds for the new values of $s_{l,j}$ and we can Lagrange interpolate $s_i = \sum_{f=1}^{3} \sum_{l\in\Omega_f\cap I_1} s_{l,i}$ similarly to the above.

To summarize, $\mathcal{S}$ can always extract s_j for $j \in I_M$. Then the ideal adversary $\mathcal{S}$ hands $(\texttt{CorruptShares}, \{j, s_j\}_{j\in I_M})$ to $\mathcal{F}_{\mathrm{KG}}$. $\mathcal{F}_{\mathrm{KG}}$ then returns $(\texttt{PublicKey}, y)$. Below we describe the computations performed by $\mathcal{S}$ before the simulation of $\mathcal{F}_{\mathrm{BB}}$ continues.

Simulation of an Honest Generator. The next problem facing the ideal adversary $\mathcal{S}$ is how to simulate the honest generators M_j for $j \notin I_M$ such that the corrupt generators M_j for $j \in I_M$ output the same public key y as that output by the dummy generators $\tilde{M}_j$ for $j \notin I_M$. The latter generators are beyond $\mathcal{S}$'s control and simply forwards the output from $\mathcal{F}_{\mathrm{KG}}$. Intuitively, $\mathcal{S}$ must "lie" at some point, since it is committed to y by the public elements on $\mathcal{F}_{\mathrm{BB}}$. The "lie" must be carefully constructed such that the adversary can not identify it, and it must be constructed not knowing $\log_g y$.

First it computes $\beta'_j = g^{s_j}$ for $j \in I_M$ and sets $\beta'_0 = y$ and $I'_M = I_M \cup \{0\}$. These are the β'_j for $j \in I_M$ that *should* later be published by the corrupted generators, but they may of course publish faulty β_j. Then $\mathcal{S}$ computes

$$\beta'_j = \prod_{i\in I'_M} (\beta'_i)^{\prod_{l\neq i} \frac{j-l}{i-l}} .$$

for $j \notin I_M$, and replace β_j with β'_j in the simulation of the honest generators M_j for $j \notin I_M$. The construction ensures that

$$\beta'_1 = \prod_{i=1}^{t+1} (\beta'_i)^{\prod_{l\neq i} \frac{l-1}{l-i}} .$$

The simulated honest parties M_j for $j \notin I_M$ are instructed not to complain if $\beta_i = \beta_i'$ for $i = 1, \ldots, t+1$.

However, it may be the case that $\beta_i' \neq \beta_i$ for $i \in I_M \cap \{1, \ldots, t+1\}$, in which case the honest generators must also simulate the handling cheating with the β_i. This is done using the same technique as above.

$\mathcal{S}$ chooses $\zeta_{j,l}$ randomly for $l \in I_M$ and sets $\zeta_{j,l} = 1$ for $j \notin I_M$. Then it replaces the original values of $\gamma_{j,l}$ and $\delta_{j,l}$ for $j \notin I_M$ by

$$\gamma_{j,l} = \prod_{i \in I_M'} \gamma_{j,i}^{\prod_{i' \neq i} \frac{l-i'}{i-i'}} .$$

This ensures that

$$g^{\zeta_{l,j}} = \prod_{\iota=0}^{t} \gamma_{l,\iota}^{j^\iota}, \quad \text{and} \quad h_{f(j)}^{\zeta_{l,j}} = \prod_{\iota=0}^{t} \delta_{l,\iota}^{j^\iota} ,$$

for $j \in I_M$. The simulated honest generators are instructed to not complain despite that the above equation does not hold for their values of $\zeta_{j,l}$, when $j, l \notin I_M$. This implies that no $\zeta_{j,l}$ for any $j \notin I_M$ is ever published.

Regardless if β_j for $j \in I_M$ are correct or not we have

$$y = \prod_{i \in I_2} \beta_i^{\prod_{l \neq i} \frac{l}{l-i}} .$$

where y is the value output by $\mathcal{F}_{\mathrm{KG}}$. At this point the simulation of $\mathcal{F}_{\mathrm{BB}}$ is continued.

Reaching a Contradiction. Suppose that $\mathcal{S}$ does not imply the security of the protocol. Then there exists a hybrid adversary $\mathcal{A}' = \mathcal{A}^{(\mathcal{S}_{\mathrm{BB}}, \mathcal{S}_{\mathrm{MMT}}, \mathcal{S}_{\mathrm{CRS}})}$, an environment $\mathcal{Z}$ with auxiliary input $z = \{z_n\}$, a constant $c > 0$ and an infinite index set $\mathcal{N} \subset \mathbb{N}$ such that for $n \in \mathcal{N}$: $|\Pr[\mathcal{Z}_z(\mathcal{I}(\mathcal{S}, \tilde{\pi}^{\mathcal{F}_{\mathrm{KG}}})) = 1] - \Pr[\mathcal{Z}_z(\mathcal{H}(\mathcal{A}', \pi^{(\tilde{\pi}_1^{\mathcal{F}_{\mathrm{BB}}}, \tilde{\pi}_2^{\mathcal{F}_{\mathrm{MMT}}}, \tilde{\pi}_3^{\mathcal{F}_{\mathrm{CRS}}})})) = 1]| \geq \frac{1}{n^c}$, where $\mathcal{S}$ runs $\mathcal{A}'$ as a black-box as described above, i.e. $\mathcal{S} = \mathcal{S}(\mathcal{A}')$.

Defining the Distinguisher. We are now ready to define a distinguisher D that contradicts the DDH assumption. D is confronted with the following test. An oracle first chooses $e_1, e_2, e_3 \in \mathbb{Z}_q$ and a bit $b \in \{0, 1\}$ randomly and defines $(u, v, w) = (h_1^{e_1}, h_1^{e_2}, h_1^{be_1e_2 + (1-b)e_3})$. Then D is given (u, v, w) and the task is to guess b.

First note that there exists $j \neq i$ such that $i, j \notin I_M$ and $f(i) \neq f(j)$. Without loss we assume that $1, 2 \notin I_M$ and $f(1) = 1$ and $f(2) = 2$. D does the following. It sets $h_2 = u$, and generates a random h_3. Then it simulates all machines and ideal functionalities as described above. However, the simulation of M_1 and M_2 is special and depends on (u, v, w).

Note that if (u, v, w) is a DDH triple, we have $(u, v, w) = (h_2, h_1^{e_2}, h_2^{e_2})$ for some random e_2. It chooses $s_{j,l}$ randomly and defines $\omega_{j,l} = h_j^{s_{j,l}}$ for $j = 1, 2$ and $l \in I_M$. Then it chooses σ randomly in $\mathbb{Z}_q$ and sets $\omega_{1,0} = h_1^{\sigma}/v$ and $\omega_{2,0} = w$.

Then it computes $s_{(1,2),l} = s_{1,l} + s_{2,l}$ for $l \in I_M$ and sets $s_{(1,2),0} = \sigma$. This allows the definition of

$$s_{(1,2),l} = \sum_{i \in I'_M} s_{(1,2),i} \prod_{j \neq i} \frac{l-i}{j-i}$$

for $l \notin I_M$. Set $s_{1,0} = \log_{h_1} \omega_{1,0}$ and $s_{2,0} = \log_{h_2} \omega_{2,0}$. These values are not known by $\mathcal{S}$, but we can still consider the equation system

$$s_{j,l} = \sum_{\iota=0}^{t} a_{j,\iota} l^{\iota}$$

for $l \in I'_M$, and find a symbolic solution $a_{j,\iota} = \sum_{l \in I'_M} d_{j,l,\iota} s_{j,l}$, for $j = 1, 2$.

D sets $\alpha_{j,\iota} = \prod_{l \in I'_M} \omega_{j,l}^{d_{j,l,\iota}}$. Then the simulation is carried through as described above, except that M_j for $j \notin I_M$ replace $s_{1,j} + s_{2,j}$ in the sum $s_j = \sum_{l \in I_1} s_{l,j}$ by $s_{(1,2),j}$ (recall that $\mathcal{S}$ does not even know $s_{1,j}$ or $s_{2,j}$).

Concluding the Proof. If (u, v, w) is a DDH triple, the distribution of the output of D is identical to the distribution of $\mathcal{Z}_z(\mathcal{H}(\mathcal{A}', \pi^{(\tilde{\pi}_1^{\mathcal{F}_{\mathrm{BB}}}, \tilde{\pi}_2^{\mathcal{F}_{\mathrm{MMT}}}, \tilde{\pi}_3^{\mathcal{F}_{\mathrm{CRS}}})}))$, whereas if (u, v, w) is not a DDH triple, the distribution is identical to the distribution of $\mathcal{Z}_z(\mathcal{I}(\mathcal{S}, \tilde{\pi}^{\mathcal{F}_{\mathrm{KG}}}))$. This implies that the DDH assumption is broken.

B Review of the UC-Security Framework

In this section we give a short review of the universally composable security framework of Canetti [3]. This framework is very general, quite complex, and hard to describe both accurately and concisely. We have chosen to use a slightly simplified approach. For a general in depth discussion, intuition, and more details we refer the reader to Canetti [3]. Note that we consider only static adversaries.

Following Goldreich [15] and Canetti [3] we define the parties to be interactive Turing machines, and denote the set of interactive Turing machines by ITM.

Canetti assumes the existence of an "operating system" that takes care of the creation of subprotocols when needed. This is necessary to handle protocols with a large number of possible trees of calls to subprotocols, but for our purposes we may assume that all subprotocols are instantiated already at the start of the protocol.

Canetti models an asynchronous communication network, where the adversary has the power to delete, modify, and insert any messages of his choice. To do this he is forced to give details for exactly what the adversary is allowed to do. This becomes quite complex in the hybrid model. We instead factor out all aspects of the communication network into a separate concrete "communication model"-machine. The real, ideal, and hybrid models are then defined solely on how certain machines are linked. The adversary is defined as any ITM, and how the adversary can interact with other machines follows implicitly from the definitions of the real and ideal communication models.

Since each protocol or subprotocol communicate through its own copy of the "communication model", and all protocols are instantiated at the start of the protocol we need not bother with session ID:s. Such ID:s would clearly be needed if our protocols would be rewritten in the more general original security framework, but it is notationally convenient to avoid them.

We also assume that we may connect any pair of machines by a "link". Such a link is more or less equivalent to the notion of a link introduced by Goldreich [15]. But the original notion of a link has the problem that it requires a machine to have a pair of communication tapes for each link, which is problematic when the number of potential links is unbounded. This is a purely definitional problem of no importance and we trust the reader to fill in the details of this in any way he or she chooses. Thus the following is meaningful.

Definition 1. *An* ITM-*graph is a set* $V = \{P_1, \ldots, P_t\} \subset \mathrm{ITM}$ *with a set of links* E *such that* (V, E) *is a connected graph, and no* P_i *is linked to any machine outside* V. *Let* ITMG *be the set of* ITM-*graphs.*

During the execution of an ITM-*graph, at most one party is* active. *An active party may deactivate itself and activate any of its neighbors, or it may halt, in which case the execution of the* ITM-*graph halts.*

The real communication model models an asynchronous communication network, in which the adversary can read, delete, modify, and insert any message of its choice. We will not make use of this communication model, but for completeness we give a definition.

Definition 2. *A* real communication model $\mathcal{C}$ *is a machine with a link* l_{P_i}, *to* P_i *for* $i = 1, \ldots, k$, *and a link* $l_{\mathcal{A}}$ *to a real adversary* $\mathcal{A}$. *Its program is defined as follows.*

1. *If* m *is read on* l_s, *where* $s \in \{P_1, \ldots, P_k\}$, *then* (s, m) *is written on* l_A *and* $\mathcal{A}$ *is activated.*
2. *If* (r, m) *is read on* $l_{\mathcal{A}}$, *where* $r \in \{P_1, \ldots, P_k\}$, *then* m *is written on* l_r, *and* r *is activated.*

The ideal communication model below captures the fact that the adversary may decide if and when it would like to deliver a message from the ideal functionality to a party, but it can not read the contents of the communication between parties and the ideal functionality.

Definition 3. *An* ideal communication model $\mathcal{C}_{\mathcal{I}}$ *is a machine with a link* l_{P_i}, *to* P_i *for* $i = 1, \ldots, k$, *and links* $l_{\mathcal{F}}$, *and* $l_{\mathcal{S}}$ *to an ideal functionality* $\mathcal{F}$ *and an ideal adversary* $\mathcal{S}$ *respectively. Its program is defined as follows.*

1. *If a message* m *is read on* l_s, *where* $s \in \{P_1, \ldots, P_k\}$, *then* (s, m) *is written on* $l_{\mathcal{F}}$ *and* $\mathcal{F}$ *is activated.*
2. *If a message* (s, m) *written on* $l_{\mathcal{F}}$ *is returned unaltered*[1], m *is written on* l_s.

[1] This special rule simplifies security proofs.

If not, any string read from $l_\mathcal{F}$ is interpreted as a list $((r_1, m_1), \ldots, (r_t, m_t))$, where $r_i \in \{\mathcal{S}, P_1, \ldots, P_k\}$. For each m_i a random string $\tau_i \in \{0,1\}^n$ is chosen, and (r_i, m_i) is stored under τ_i. Then $((r_1, |m_1|, \tau_1), \ldots, (r_t, |m_t|, \tau_t))$, where $|m_i|$ is the bit-length of m_i, is written to $l_\mathcal{S}$ and $\mathcal{S}$ is activated.

3. *Any string read from $l_\mathcal{S}$ is interpreted as a pair (b, τ), where $b \in \{0,1\}$ and τ is an arbitrary string. If $b = 1$ and (r_i, m_i) is stored in the database under the index τ, m_i is written on l_{r_i} and r_i is activated. Otherwise $(\mathcal{S}, \tau)$ is written to $l_\mathcal{F}$ and $\mathcal{F}$ is activated.*

An *adversary* can normally corrupt some subset of the parties in a protocol. A *dummy party* is a machine that given two links writes any message from one of the links on the other. There may be many copies of the dummy party. Following Canetti we use the $\tilde{\ }$-notation, e.g. $\tilde{P}$, for dummy parties.

The ideal model below captures the setup one wishes to realize, i.e. the environment may interact with the ideal functionality $\mathcal{F}$, except that the adversary $\mathcal{S}$ has some control over how the communication model behaves.

Definition 4. *The* ideal model *is defined to be a map* $\mathcal{I} : \mathrm{ITM}^2 \times \tilde{\mathrm{ITM}}^* \rightarrow \mathrm{ITMG}$*, where* $\mathcal{I} : (\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k) \mapsto (V, E)$ *is given by:*

$$V = \{\mathcal{C}_\mathcal{I}, \mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k\}, \quad E = \{(\mathcal{S}, \mathcal{C}_\mathcal{I}), (\mathcal{C}_\mathcal{I}, \mathcal{F})\} \cup \bigcup_{i=1}^{k} \{(\tilde{P}_i, \mathcal{C}_\mathcal{I})\} \ .$$

If $\tilde{\pi} = (\tilde{P}_1, \ldots, \tilde{P}_k)$, we write $\mathcal{I}(\mathcal{S}, \tilde{\pi}^\mathcal{F})$ instead of $\mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k)$ to ease notation.

The real model is supposed to capture the properties of the real world. The parties may interact over the real communication model.

Definition 5. *The* real model *is defined to be a map* $\mathcal{R} : \mathrm{ITM}^* \rightarrow \mathrm{ITMG}$*, where* $\mathcal{R} : (\mathcal{A}, P_1, \ldots, P_k) \mapsto (V, E)$ *is given by:*

$$V = \{\mathcal{C}, \mathcal{A}, P_1, \ldots, P_k\}, \quad E = \{(\mathcal{A}, \mathcal{C})\} \cup \bigcup_{i=1}^{k} \{(P_i, \mathcal{C})\} \ .$$

Let $(V, E) = \mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k)$. Then we write $\mathcal{Z}(\mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k))$ for the ITM-graph (V', E') defined by $V' = V \cup \{\mathcal{Z}\}$, and $E' = E \cup \{(\mathcal{Z}, \mathcal{S})\} \cup \bigcup_{i=1}^{k} \{(\mathcal{Z}, \tilde{P}_i)\}$. We use the corresponding notation in the real model case.

A hybrid model is a mix between a number of ideal and real models, and captures the execution of a real world protocol with access to some ideal functionalities. It is also a tool to modularize security proofs. It may be viewed as if we "glue" a number of ideal and real models onto an original real model.

Definition 6. *Suppose that we are given* $(V, E) = \mathcal{R}(\mathcal{A}, \pi)$*,* $\pi = (P_1, \ldots, P_k)$*. Let* $(V_j, E_j) = \mathcal{I}(\mathcal{S}_j, \tilde{\pi}_j^{\mathcal{F}_j})$*,* $\tilde{\pi}_j = (\tilde{P}_{j,1}, \ldots, \tilde{P}_{j,k})$ *for* $j = 1, \ldots, t$*, and* $(V_j, E_j) = \mathcal{R}(\mathcal{S}_j, \pi_j)$*,* $\pi_j = (P_{j,1}, \ldots, P_{j,k})$ *for* $j = t + 1, \ldots, s$*.*

We denote by $\mathcal{H}(\mathcal{A}^{(\mathcal{S}_1,\ldots,\mathcal{S}_t)}, \pi^{(\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t},\pi_{t+1},\ldots,\pi_s)})$ *the* hybrid model *defined as the* ITM-*graph* (V', E')*, where*

$$V' = V \cup \bigcup_{j=1}^{t} V_j, \ \ \textit{and} \ \ E' = E \cup \bigcup_{j=1}^{t} E_j \cup \bigcup_{i=1}^{k} \left(\{(\mathcal{S}_i, \mathcal{A})\} \cup \bigcup_{j=1}^{t} \{(P_i, \tilde{P}_{j,i})\} \right) \ .$$

Similarly as above we write $\mathcal{Z}(\mathcal{H}(\mathcal{A}^{(\mathcal{S}_1,\ldots,\mathcal{S}_t)}, \pi^{(\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t},\pi_{t+1},\ldots,\pi_s)}))$ to denote the ITM-graph (V'', E'') defined by $V'' = V' \cup \{\mathcal{Z}\}$, and $E'' = E' \cup \{(\mathcal{Z}, \mathcal{A})\} \cup \bigcup_{i=1}^{k} \{(\mathcal{Z}, P_i)\}$.

Note that all real subprotocols π_j, for $j = t+1, \ldots, s$, above may be integrated into the original real protocol π. Thus a hybrid model with no ideal functionalities involved is equivalent to a real model, except that it may use several communication models. One may either augment the definition of a real model to allow this, or only use communication models with the property that two communication models can be simulated using a single communication model. The real communication model above, Definition 2, has this property.

The concept of hybrid models is generalized in the natural way, e.g. we write $\mathcal{H}(\mathcal{A}^{(A_1^{\mathcal{S}_{11}}, A_2^{\mathcal{S}_{21}})}, \pi^{(\pi_1^{\tilde{\pi}_{11}^{\mathcal{F}}}, \pi_2^{\tilde{\pi}_{21}^{\mathcal{F}}})})$ for a hybrid model for a real protocol that executes two subprotocols, where each subprotocol has access to a separate copy of the ideal functionality $\mathcal{F}$. Some care needs to be taken when defining the adversary for such models. If an adversary corrupts a party, it automatically corrupts all its sub-parties that are involved in subprotocols[2].

We also write $\mathcal{Z}_z$ to denote that $\mathcal{Z}$ takes auxiliary input z, and always assume that in any execution of such an ITM-graph, $\mathcal{Z}$ is activated first.

The following definition is somewhat sloppy in that we have not defined the notion of $\mathcal{M}$-adversaries rigorously. We trust the reader to resolve this, and assume that $\mathcal{M}$ is some class of adversaries.

Definition 7 (Secure Realization). *Let* $\mathcal{F}$ *be an ideal functionality. Let* $\pi = (P_1, \ldots, P_k)$*, and let* $\tilde{\pi}_j = (\tilde{P}_{j,i}, \ldots, \tilde{P}_{j,i})$ *be the corresponding dummy parties for* $\mathcal{F}_j$*, for* $j = 1, \ldots, t$*.*

Then $\pi^{(\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t})}$ *realizes* $\tilde{\pi}^{\mathcal{F}}$ *securely with regards to* $\mathcal{M}$*-adversaries if for all* $\mathcal{M}$*-adversaries* $\mathcal{A}^{(S_1,\ldots,S_t)}$ *with auxiliary input* $z = \{z_n\}$*,* $\exists \mathcal{S} \in \mathrm{ITM}$ *such that* $\forall c > 0$*,* $\exists n_0$*, such that* $\forall n > n_0$

$$|\Pr[\mathcal{Z}_z(\mathcal{I}(\mathcal{S}, \tilde{\pi}^{\mathcal{F}})) = 1] - \Pr[\mathcal{Z}_z(\mathcal{H}(\mathcal{A}^{(S_1,\ldots,S_t)}, \pi^{(\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t})})) = 1]| < \frac{1}{n^c} \ .$$

Since the dummy parties are of no real importance we also say that π *realizes* $\mathcal{F}$ *in the* $(\mathcal{F}_1, \ldots, \mathcal{F}_t)$*-hybrid model.*

Canetti [3] proves a powerful composition theorem that can handle polynomially many instances of a constant number of ideal functionalities, but we only need the following weaker special case.

[2] The most general definition allows some violations of this rule.

Theorem 2 (Composition Theorem). *Suppose that* $\pi^{(\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t})}$ *securely realizes* $\tilde{\pi}^{\mathcal{F}}$*, and that* $\pi_i^{(\tilde{\pi}_{i1}^{\mathcal{F}_{i1}},\ldots,\tilde{\pi}_{it_i}^{\mathcal{F}_{it_i}})}$ *securely realizes* $\tilde{\pi}_i^{\mathcal{F}_i}$*, for* $i = 1, \ldots, l$*, with regards to* $\mathcal{M}$*-adversaries.*

Then $\pi^{(\pi_1^{(\tilde{\pi}_{11}^{\mathcal{F}_{11}},\ldots,\tilde{\pi}_{1t_1}^{\mathcal{F}_{1t_1}})},\ldots,\pi_l^{(\tilde{\pi}_{l1}^{\mathcal{F}_{l1}},\ldots,\tilde{\pi}_{lt_l}^{\mathcal{F}_{lt_l}})},\tilde{\pi}_{l+1}^{\mathcal{F}_{l+1}},\ldots,\tilde{\pi}_t^{\mathcal{F}_t})}$ *securely realizes* $\tilde{\pi}^{\mathcal{F}}$ *with regards to* $\mathcal{M}$*-adversaries.*

Proof. A full proof of the more general statement can be found in Canetti [3].

Note that the hybrid protocol can be transformed into a protocol in the $(\mathcal{F}_{11}, \ldots, \mathcal{F}_{1t_1}, \ldots, \mathcal{F}_{l1}, \ldots, \mathcal{F}_{lt_l}, \mathcal{F}_{l+1}, \ldots, \mathcal{F}_t)$-hybrid model. However, in this hybrid model the underlying real model may use several different communication models.